

REMARKS

Claims 1-26 are pending. By this amendment, Claims 1-5, 12-16, 23, 24, and 26 are cancelled without prejudice to their reintroduction in a continuation. Reconsideration is respectfully requested.

It is noted that the claims fall into three logical sets. Speaking only of the independent claims, the claims can be grouped as: 1) Claims 1, 12, 23, and 26 translate source code into a markup language; 2) Claims 5, 16, and 24 translate a markup language into source code; and 3) Claims 6, 17, and 25 perform a dynamic translation of a programming language application into markup language. Of these, the claims from the first two sets are cancelled, while the third set, to a dynamic translation, remains.

Rejections under §103

Claims 5-11, 16-22, and 24-25 are rejected under 103(a) as unpatentable over Meltzer. Of these, Claims 5, 16, and 24 are cancelled with out prejudice, and their rejection is moot.

Claim 6 is representative of the claims to a dynamic translation of an application program into a markup language. This claim reads,

6. (Amended) A method of dynamically translating an application program into a markup language file, the method comprising the computer-implemented steps of:
- executing said application program;
 - parsing a document type definition file for a markup language;
 - during execution of said application program, selecting an element defined in the document type definition file based on a routine called by said application program; and
 - writing the selected element to a markup language file to form a translation.

Note that in this claim, the preamble states that an application program will be translated into markup language; in the body of the claim, that same application program must be executed in order to be translated. Note further, the claim does not say that the application program is performing the translation; rather it is the object of the translation. The "selecting" and "writing" steps, which are being performed in tandem with the execution, are making the actual translation.

The rejection states, regarding the above Claim 6,

"executing an application program (Meltzer on col. 23, lines 17-60); teaches the objects would be transformed into format required by the receiving

application; ... during execution of said program application (on col. 23, lines 17-60): teaches running listeners as JAVA functions); selecting an element defined in the document type definition file based on a routine called by the application program (Meltzer on col. 23, lines 17-60): teaches element retrieved from XML DTD"¹

Applicants submit that there are at least two problems with this rejection: a) the rejection does not address the correspondence between the claim and Meltzer in a manner that is clearly understood and b) the patent does not perform a dynamic translation. We shall discuss these in turn.

a) The rejection does not clearly address the correspondence between the claim and Meltzer, although it attempts to do so. For example, the rejection does not make clear what it considers to be the claimed application program that is being translated. Is the rejection equating the claimed application program to the objects that will be transformed, to the receiving application, or to the listeners? What corresponds to the claimed routine that the application program calls? **If the examiner persists in her rejection, she is respectfully requested to clarify these points so that applicant can more specifically direct their response.**

b) Meltzer does not perform a dynamic translation and does not meet the claim limitations. The rejection repeatedly cites Meltzer, column 23, lines 17-60, which state,

"Furthermore, according to the present invention the applications that the listeners run need not be native XML functions, or native functions which match the format of the incoming document. Rather, these listeners may be JAVA functions, if the transaction process front end 304 is a JAVA interface, or may be functions which run according to a unique transaction processing architecture. In these cases, the objects would be transformed into the format required by the receiving application. When the application of the listener finishes, its output is then transformed back into the format of a document as specified by the business interface definition in the module 303. Thus, the translator 302 is coupled to the network interface 300 directly for supplying the composed documents as outputs.

The listeners coupled to the transaction processing front end may include listeners for input documents, listeners for specific elements of the input documents, and listeners for attributes stored in particular elements of the input document. This enables diverse and flexible implementations of transaction processes at the participant nodes for filtering and responding to incoming documents.

FIG. 4 illustrates a process of receiving and processing an incoming document for the system of FIG. 3. Thus, the process begins by receiving a document at the network interface (step 400). The parser identifies the document type (401) in response to the business interface definition. Using the business interface definition, which stores a DTD for the document in the

¹ Office Action of 4/09/03, page 4, lines 7-12

XML format, the document is parsed (step 402). Next, the elements and attributes of the document are translated into the format of the host (step 403). In this example, the XML logic structures are translated into JAVA objects which carry the data of the XML element as well as methods associated with the data such as get and set functions. Next, the host objects are transferred to the host transaction processing front end (step 404). These objects are routed to processes in response to the events indicated by the parser and the translator. The processes which receive the elements of the document are executed and produce an output (step 405). The output is translated to the format of an output document as defined by the business interface definition (step 406). In this example, the translation proceeds from the form of a JAVA object to that of an XML document. Finally, the output document is transmitted to its destination through the network interface (step 407).²

Applicants submit that there is no discussion in this section of executing an application in order to effect the translation of that application, yet this is essential to the claimed dynamic translation. It is further submitted that the claim specifically recites that the application program calls a routine and that an element is chosen based on that routine. Yet Meltzer neither discusses that the application program calls a routine nor that an element is chosen in correspondence to this routine.

It is further submitted that while the extract above mentions that XML structures are translated into JAVA objects and JAVA objects into XML documents, there is no indication of how these are translated. More specifically, these translations do not appear to be performed while the program to be translated is executing. Thus, although Meltzer may show some type of translation, Meltzer does not show a dynamic translation. This rejection is overcome.

Claims 1-4, 12-15, 23 and 26 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Meltzer in view of Wu et al. (5,987,256).

These claims have been cancelled without prejudice, so this rejection is moot.

² Meltzer, column 23, lines 17-60

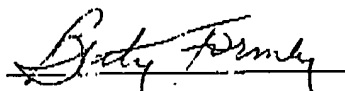
Conclusion

It is respectfully urged that the subject application is patentable over the references relied on and is now in condition for allowance.

The Examiner is requested to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: July 3, 2003

Respectfully submitted,



Betty Fornby
Reg. No. 36,536
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
Agent for Applicants